CLAIMS

1. A method for dynamically patching code, comprising the steps of:

intercepting program instructions;

determining if a program instruction requires unavailable hardware functionality; and

dynamically replacing the program instruction with a replacement instruction that does not require unavailable hardware functionality if it is determined that the program instruction requires unavailable hardware functionality.

2. The method of claim 1, wherein the step of dynamically replacing the program instruction comprises fetching a replacement instruction and storing it in a code cache.

3. The method of claim 2, wherein the step of dynamically replacing the program instruction further comprises executing the replacement instruction in lieu of the program instruction each time a function associated with the program instruction is required.

4. The method of claim 3, wherein the replacement instruction comprises part of a patch that is made available via an application programming interface.

5. The method of claim 1, further comprising the step of, prior to determining if a program instruction requires unavailable hardware functionality, determining if the program instruction has been cached.

6. The method of claim 5, further comprising the step of executing the cached instruction in lieu of the program instruction if an associated instruction has been cached.

7. The method of claim 1, further comprising the step of, prior to intercepting program instructions, gaining control over execution of program instructions by injecting a dynamic execution layer interface into the program.

8. The method of claim 1, further comprising the step of dynamically receiving information about unavailable hardware functionality and replacement instructions that are configured to replace original program instructions that require the unavailable hardware functionality.

9. A system for dynamically patching code, comprising:

means for gaining control over execution of a program;

means for intercepting program instructions;

means for determining if a program instruction requires unavailable hardware functionality; and

means for dynamically replacing the program instruction with a replacement instruction that does not require unavailable hardware functionality if it is determined that the program instruction requires unavailable hardware functionality.

10. The system of claim 9, wherein the means for dynamically replacing the program instruction comprise means for fetching a replacement instruction and storing it in a code cache.

11. The system of claim 9, further comprising means for determining if a program instruction has been cached.

12. The system of claim 9, further comprising means for dynamically receiving information about unavailable hardware functionality and replacement instructions that are configured to replace original program instructions that require the unavailable hardware functionality.

13. A dynamic patching program stored on a computer-readable medium, comprising:

logic configured to gain control over execution of a program;

logic configured to intercept program instructions;

logic configured to determine if a program instruction requires unavailable hardware functionality; and

logic configured to dynamically replace the program instruction with a replacement instruction that does not require unavailable hardware functionality if it is determined that the program instruction requires unavailable hardware functionality.

14. The system of claim 13, wherein the logic configured to dynamically replace the program instruction comprises logic configured to fetch a replacement instruction and store it in a code cache.

15. The system of claim 13, further comprising logic configured to determine if a program instruction has been cached.

16. The system of claim 13, further comprising logic configured to dynamically receive information about unavailable hardware functionality and replacement instructions that are configured to replace original program instructions that require the unavailable hardware functionality.

17. A method for dynamically patching code, comprising the steps of:

gaining control over the execution of a program;

intercepting program instructions;

determining whether the program instructions have been cached and, if so, executing the cached instructions;

if the program instructions have not been cached, determining if the program instructions require unavailable hardware functionality; and

dynamically replacing the program instructions with replacement instructions that do not require unavailable hardware functionality if it is determined that the program instructions require unavailable hardware functionality.

18. The method of claim 17, wherein the step of dynamically replacing the program instructions comprises fetching replacement instructions and storing them in a code cache.

19. The method of claim 18, wherein the step of dynamically replacing the program instructions further comprises executing the replacement instructions in lieu of the program instructions each time a functionality associated with the program instructions is required.

20. The method of claim 19, wherein the replacement instructions comprise part of a patch that is made available via an application programming interface.

21. A dynamic execution layer interface (DELI) residing between an application and computing system hardware, comprising:

a transparent mode layer that is configured to gain control over the operation of the application and to fetch replacement instructions that are to replace existing application instructions;

a system control and configuration layer configured to provide policies for the replacement of existing application instructions with the replacement instructions;

a core configured to dynamically cache and execute the replacement instructions; and

a code cache in which the replacement instructions are cached.

22. The DELI of claim 21, wherein the transparent mode layer is further configured to fetch application instructions from the application and wherein the core is further configured to cache fetched application instructions in the code cache.